Indian Institute of Technology Kanpur

CS335: Compiler Design

Course Project

Milestone II

2024

Group 34
Rohan Ravi(210870)
Aryan Maurya(210202)
Narendra Singh(210649)

0.1. Compilation and Execution

- In the provided milestone2.zip folder, you will find the following files:
  - src
    - lexer.l
    - parser.y
    - build.sh : make sure it has execute permissions. Use to generate 'compiler.exe'
    - clean.sh : make sure it has execute permissions. Use to clean out generated files.
  - doc
    - readme.pdf
  - test
    - exp.py : to test out expressions, assignments, variable declarations etc.
    - conditional.py : to test conditional jumps if-elif-else statements
    - loops.py : while, for loops along with break and continue statements
    - func.py : function definitions and calls
    - class.py: class definitions with inheritance, methods and method overloading.
- ./build.sh creates compiler.exe
- The following are the flags used for executions:
  - $-$**input** : provide path to testcase. Eg; $-input../test/class.py$
  - $-$**sym** : generates symboltables.csv
  - $-$**3ac**: generates 3AC code in 3AC.txt
  - $-$**ast** to create AST.dot of the parse tree.
  - $--$**debug**: [FOR DEBUGGING] prints out all non-terminals parsed.
  - $--$**st**: [FOR DEBUGGING] prints out symbol table in the terminal.
- eg: ./compiler.exe -input ../test/class.py -sym -3ac
- You will find 2 files 'symboltables.csv' and '3AC.txt' in the current directory.
- **symboltables.csv** contains csv for all symbol tables **in the same csv file**, including information like TableName, ParentTable and ChildTables, followed by each entry
- **3AC.txt** is the 3AC for the given input python program.

0.2. Milestone 2

The following is a brief:

- Symbol Table is maintained by extracting relevant information from declarations. Info like type, function signature, line no, offsets etc.
  Symbol Table supports operations like entry() for insertion and lookup() used for error checking.
  Hierarchical 2-level symbol table structure is followed.

- Semantic Analysis: following are a few features explained. More can be found while testing using our testcases and during TA evaluation.
  - scope checking is done using symbol table structure as explained earlier.
  - Type correctness is checked.
  - Function argument match is done by storing function signature in the symbol table itself.
  - Object Oriented programming support like method inheritance and Static polymorphism with method/function overloading.
- 3AC code is generated with the following 3AC instructions defined: pushparam, popparam, goto, ifz, call along with Labels.
- predefined functions range(start,stop,step), print() have supported 3AC conversion.
  Support for len() will be added later.